IN THE CLAIMS:

Please amend claims 1 and 25 as indicated in the following.

Claims Listing:

1. (Currently Amended) A method comprising:

receiving a graphics function call at a driver;

converting, at the driver, the graphics function call to a native command set for execution on a native system; and

capturing the native command set in a database for subsequent performance analysis.

2. (Original) The method as in Claim 1, wherein the database includes a single file.

3. (Canceled)

4. (Original) The method as in Claim 1, wherein the graphics function call is a draw command.

5. (Original) The method as in Claim 4, wherein when the draw command is an indexed draw command, capturing the native command set includes storing the indexed draw command and storing a direct draw command associated with the indexed draw command.

6. (Original) The method as in Claim 5, wherein, for each captured indexed draw command, a first mode of operation executes a direct draw command related to the indexed draw command directly and a second mode of operation stores data associated with the direct draw command in buffer locations and then executes the indexed draw command.

7. (Original) The method as in Claim 5, wherein stored indexed draw commands are made inoperative using a no-op command.

8. (Original) The method as in Claim 7, wherein the no-op command is a type-3 no-op command.

9. (Original) The method as in Claim 7, wherein, for each captured indexed draw command, a first mode of operation executes a direct draw command related to the indexed draw command directly and a second mode of operation stores data associated with the direct draw command in buffer locations and then executes the indexed draw command.

10. (Original) The method as in Claim 5, wherein storing indexed draw commands includes storing vertex data associated with the indexed draw command.

11. (Previously Presented) The method as in Claim 10, further including:
configuring the driver to provide a virtual address indicating a location of the vertices of the indexed draw command when the indexed draw command uses a physical location of the vertices.

12. (Previously Presented) The method as in Claim 1, further including:
issuing the native command set to a first system, capable of executing the native command set, simultaneous with the capturing of the native command set.

13. (Previously Presented) The method as in Claim 12, further including:
providing commands in the database to a second system for execution subsequent to the capturing of the native command set.

14. (Original) The method as in Claim 13, wherein the first system and the second system are the same system.

15. (Previously Presented) The method as in Claim 14, wherein the first system and the second system include at least one of: a graphics chip; a hardware emulator; a software simulator; or an architectural description analyzer.

16. (Original) The method as in Claim 13, wherein the first system and the second system are different systems.

17. (Previously Presented) The method as in Claim 1, further including:

re-mapping a system to match indexed physical values, wherein the system is used for simulating a native system related to the native command set.

18. (Previously Presented) The method as in Claim 17, further including:

simulating the native system by executing one or more commands from the native command set.

19. (Previously Presented) The method as in Claim 18, further including:

tracking the number of clock cycles in executing the one or more commands.

20. (Previously Presented) The method as in Claim 18, further including:

un-mapping the system when the simulation is complete.

21. (Previously Presented) The method as in Claim 1, further including:

receiving architectural description; and

determining an estimated performance based on the architectural description and workload characteristics of the native command set.

22. (Original) The method as in Claim 21, wherein the workload characteristics relate to 3D graphics performance, as defined with the native command set.

23. (Original) The method as in Claim 21, wherein the workload characteristics relate to 2D graphics performance, as defined with the native command set.

24. (Original) The method as in Claim 21, wherein the workload characteristics relate to memory usage, as defined with the native command set.

25. (Currently Amended) A system comprising:

a data processor having an I/O buffer; and

a memory having an I/O buffer coupled to the I/O buffer of the data processor; the memory capable of storing code for:

an application capable of generating graphics function calls;

a driver capable of capturing the graphics function calls and converting the graphics function calls to a native command set; and

a routine capable of capturing the native command set to a database for subsequent performance analysis.

26. (Original) The system as in Claim 25, wherein the database includes a single file for collecting the native command set.

27. (Original) The system as in Claim 25, wherein the driver includes a queue server, wherein a queue server is capable of receiving multiple commands of the native command set and outputting the commands.

28. (Original) The system as in Claim 25, wherein the graphics function call is a draw command.

29. (Original) The system as in Claim 28, wherein, for each indexed draw command, the routine is further capable of storing the indexed draw command and storing a direct draw command associated with the indexed draw command.

30. (Original) The system as in Claim 29, wherein, for each stored indexed draw command, a first mode of operation executes the direct draw command associated with the indexed draw command directly and a second mode of operation stores data associated with direct draw command in buffer locations and then executes the indexed draw command.

31. (Original) The system as in Claim 29, wherein stored indexed draw commands are made inoperative using a no-op command.

32. (Original) The system as in Claim 31, wherein the no-op command is a type-3 no-op command.

33. (Original) The system as in Claim 31, wherein, for each stored indexed draw command, a first mode of operation executes the direct draw command associated with the indexed draw command directly and a second mode of operation stores data associated with direct draw command in buffer locations and then executes the indexed draw command.

34. (Original) The system as in Claim 29, wherein storing indexed draw commands includes storing vertex data associated with the indexed draw commands.

35. (Original) The system as in Claim 34, wherein the driver is configured to provide a virtual address indicating a location of the vertices of the indexed draw command when the indexed draw command uses a physical location of the vertices.

36. (Previously Presented) The system as in Claim 25, further including:
a first system capable of executing the native command set simultaneous with the capturing of the native command set.

37. (Previously Presented) The system as in Claim 36, further including:
a second system capable of executing the native command set captured in the database.

38. (Original) The system as in Claim 37, wherein the first system and the second system are the same system.

39. (Previously Presented) The system as in Claim 38, wherein the first system and the second system include at least one of: a graphics chip; a hardware emulator; a software simulator; or an architectural description analyzer.

40. (Original) The system as in Claim 37, wherein the first system and the second system are different systems.

41. (Original) The system as in Claim 37, wherein the second system is re-mapped to match indexed physical values, and wherein the second system is used for simulating a native system related to the native command set.

42. (Previously Presented) The system as in Claim 41, wherein the second system is further capable of simulating the native system by executing one or more commands of the native command set.

43. (Previously Presented) The system as in Claim 42, wherein the second system includes counters capable of tracking the number of clock cycles used in executing the one or more commands.

44. (Original) The system as in Claim 42, wherein the second system is un-mapped when the simulation is complete.

45. (Previously Presented) The system as in Claim 25, further including:
an architectural description; and
a performance analyzer capable of determining an estimated performance based on the architectural description and workload characteristics.

46. (Original) The system as in Claim 45, wherein the workload characteristics relate to 3D graphics performance, as defined with the native command set.

47. (Original) The system as in Claim 45, wherein the workload characteristics relate to 2D graphics performance, as defined with the native command set.

48. (Original) The system as in Claim 45, wherein the workload characteristics relate to memory usage, as defined with the native command set.

49. (Previously Presented) A method of determining graphics processing performance, the method comprising:

receiving a graphics function call at a driver;

converting, at the driver, the graphics function call to a native command set for execution on a native system;

capturing the native command set in a database; and

executing the command set to determine graphics processing performance.

50. (Original) The method as in Claim 49, wherein the database includes a single file.

51. (Original) The method as in Claim 49, wherein executing includes hardware emulation.

52. (Original) The method as in Claim 49, wherein executing includes software simulation.

53. (Original) The method as in Claim 49, wherein the driver includes a queue server, wherein the queue server is capable of receiving multiple commands of the native command set and outputting the commands.

54. (Original) The method as in Claim 49, wherein the graphics function call is a draw command.

55. (Original) The method as in Claim 54, wherein, for each indexed draw command, capturing the native command set includes storing the indexed draw command and storing a direct draw command associated with the indexed draw command.

56. (Original) The method as in Claim 55, wherein, for each stored indexed draw command, a first mode of operation executes the direct draw command associated with the indexed draw command directly and a second mode of operation stores data associated with direct draw command in buffer locations and then executes the indexed draw command.

57. (Original) The method as in Claim 55, wherein stored indexed draw commands are made inoperative using a no-op command.

58. (Original) The method as in Claim 57, wherein the no-op command is a type-3 no-op command.

59. (Original) The method as in Claim 57, wherein, for each stored indexed draw command, a first mode of operation executes the direct draw command associated with the indexed draw command directly and a second mode of operation stores data associated with direct draw command in buffer locations and then executes the indexed draw command.

60. (Original) The method as in Claim 55, wherein storing indexed draw commands includes storing vertex data associated with the indexed draw commands.

61. (Previously Presented) The method as in Claim 60, further including:

configuring the driver to provide a virtual address and to request a physical address indicating a location of the vertices of the indexed draw command when the indexed draw command uses a physical location of the vertices.

62. (Previously Presented) The method as in Claim 49, further including:

issuing the native command set to a first system, capable of executing the native command set, simultaneous with the capturing of the native command set.

63. (Previously Presented) The method as in Claim 62, further including:

providing commands in the single file to a second system for execution subsequent the capturing of the native command set.

64. (Original) The method as in Claim 63, wherein the first system and the second system are the same system.

65. (Previously Presented) The method as in Claim 64, wherein the first system and the second system include at least one of: a graphics chip; a hardware emulator; a software simulator; or an architectural description analyzer.

66. (Original) The method as in Claim 63, wherein the first system and the second system are different systems.

67. (Previously Presented) The method as in Claim 49, further including:
re-mapping a system to match indexed physical values, wherein the system is used for simulating a native system related to the native command set.

68. (Previously Presented) The method as in Claim 67, further including:
simulating the native system by executing one or more commands of the native command set.

69. (Previously Presented) The method as in Claim 68, further including:
tracking the number of clock cycles in executing the one or more commands.

70. (Previously Presented) The method as in Claim 68, further including:
un-mapping the system when the simulation is complete.

71. (Previously Presented) The method as in Claim 49, further including:
receiving architectural description; and
determining an estimated performance based on the architectural description and workload characteristics of the native command set.

72. (Original) The method as in Claim 71, wherein the workload characteristics relate to 3D graphics performance, as defined with the native command set.

73. (Original) The method as in Claim 71, wherein the workload characteristics relate to 2D graphics performance, as defined with the native command set.

74. (Original) The method as in Claim 71, wherein the workload characteristics relate to memory usage, as defined with the native command set.